

COL702: Advanced Data Structures and Algorithms

Ragesh Jaiswal, CSE, IITD

Applications of Network Flow

- Suppose there are four teams in IPL with their current number of wins:
 - Daredevils: 10
 - Sunrisers: 10
 - Lions: 10
 - Supergiants: 8
- There are 7 more games to be played. These are as follows:
 - Supergiants plays all other 3 teams.
 - Daredevils Vs Sunrisers, Sunrisers Vs Lions, Daredevils Vs Lions, Sunrisers Vs Daredevils

- Suppose there are four teams in IPL with their current number of wins:
 - Daredevils: 10
 - Sunrisers: 10
 - Lions: 10
 - Supergiants: 8
- There are 7 more games to be played. These are as follows:
 - Supergiants plays all other 3 teams.
 - Daredevils Vs Sunrisers, Sunrisers Vs Lions, Daredevils Vs Lions, Sunrisers Vs Daredevils
- A team is said to be eliminated if it cannot end with maximum number of wins.
- Can we say that Supergiants have been eliminated give the current scenario?

- Suppose there are four teams in IPL with their current number of wins:
 - Daredevils: 10
 - Sunrisers: 10
 - Lions: 9
 - Supergiants: 8
- There are 7 more games to be played. These are as follows:
 - Supergiants plays all other 3 teams.
 - 4 games between Daredevils and Sunrisers.
- Can we say that Supergiants have been eliminated give the current scenario?

Problem

There are n teams. Each team i has a current number of wins denoted by $w(i)$. There are $G(i, j)$ games yet to be played between team i and j . Design an algorithm to determine whether a given team x has been eliminated.

Network Flow

Team Elimination

Problem

There are n teams. Each team i has a current number of wins denoted by $w(i)$. There are $G(i, j)$ games yet to be played between team i and j . Design an algorithm to determine whether a given team x has been eliminated.

- Consider the following flow network

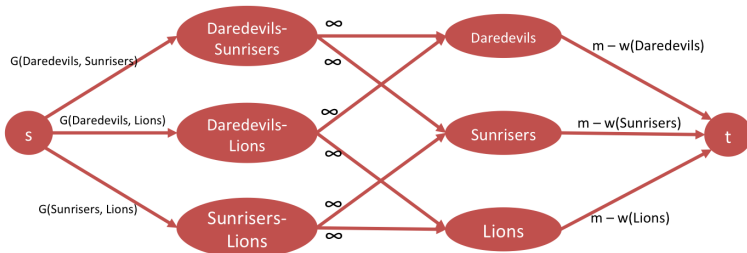


Figure: Team x can end with at most m wins, i.e., $m = w(x) + \sum_j G(x, j)$

Network Flow

Team Elimination

Problem

There are n teams. Each team i has a current number of wins denoted by $w(i)$. There are $G(i, j)$ games yet to be played between team i and j . Design an algorithm to determine whether a given team x has been eliminated.

- Claim 1: Team x has been eliminated **iff** the maximum flow in the network is $< g^*$, where $g^* = \sum_{i,j \text{ s.t. } x \notin \{i,j\}} G(i, j)$.

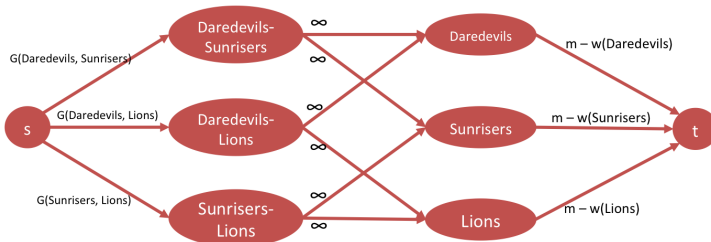


Figure: Team x can end with at most m wins, i.e., $m = w(x) + \sum_j G(x, j)$

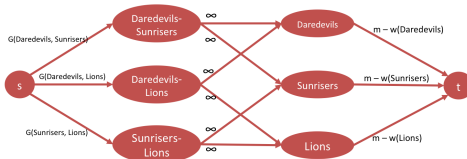
Network Flow

Team Elimination

Problem

There are n teams. Each team i has a current number of wins denoted by $w(i)$. There are $G(i, j)$ games yet to be played between team i and j . Design an algorithm to determine whether a given team x has been eliminated.

- Claim 1: Team x has been eliminated **iff** the maximum flow in the network is $< g^*$, where $g^* = \sum_{i, j \text{ s.t. } x \notin \{i, j\}} G(i, j)$.
- Comment: If we can somehow find a subset T of teams (not including x) such that $\sum_{i \in T} w(i) + \sum_{i < j \text{ and } i, j \in T} G(i, j) > m \cdot |T|$. Then we have a witness to the fact that x has been eliminated.



Network Flow

Team Elimination

Problem

There are n teams. Each team i has a current number of wins denoted by $w(i)$. There are $G(i, j)$ games yet to be played between team i and j . Design an algorithm to determine whether a given team x has been eliminated.

- Claim 1: Team x has been eliminated **iff** the maximum flow in the network is $< g^*$, where $g^* = \sum_{i,j \text{ s.t. } x \notin \{i,j\}} G(i, j)$.
- Comment: If we can somehow find a subset T of teams (not including x) such that $\sum_{i \in T} w(i) + \sum_{i < j \text{ and } i, j \in T} G(i, j) > m \cdot |T|$. Then we have a witness to the fact that x has been eliminated.
- Can we find such a subset T ?



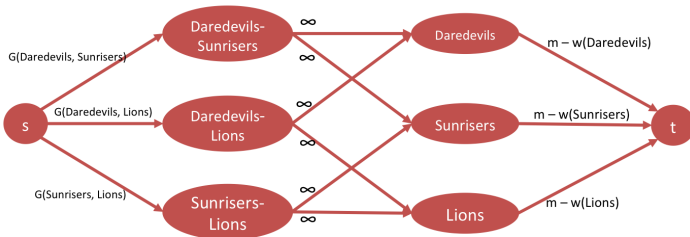
Network Flow

Team Elimination

- Claim 1: Team x has been eliminated **iff** the maximum flow in the network is $< g^*$, where $g^* = \sum_{i,j \text{ s.t. } x \notin \{i,j\}} G(i,j)$.

Proof.

- Claim 1.1: If x has been eliminated, then the max flow in the network is $< g^*$.



Network Flow

Team Elimination

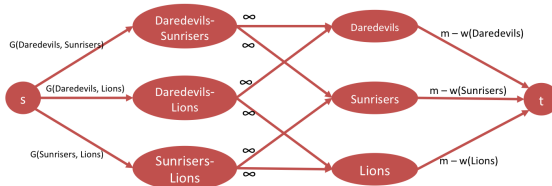
- Claim 1: Team x has been eliminated **iff** the maximum flow in the network is $< g^*$, where $g^* = \sum_{i,j \text{ s.t. } x \notin \{i,j\}} G(i,j)$.

Proof of Claim 1

- Claim 1.1: If x has been eliminated, then the max flow in the network is $< g^*$.
- Claim 1.2: If the max flow is $< g^*$, then team x has been eliminated.

Proof of Claim 1.2

- Consider any s - t min-cut (A, B) in the graph.
- Claim 1.2.1: If v_{ij} is in A , then both v_i and v_j are in A .



Network Flow

Team Elimination

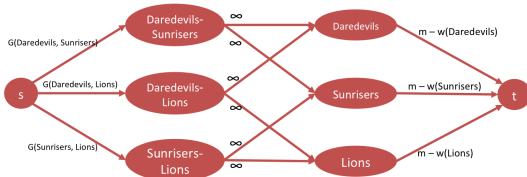
- Claim 1: Team x has been eliminated **iff** the maximum flow in the network is $< g^*$, where $g^* = \sum_{i,j \text{ s.t. } x \notin \{i,j\}} G(i,j)$.

Proof of Claim 1

- Claim 1.1: If x has been eliminated, then the max flow in the network is $< g^*$.
- Claim 1.2: If the max flow is $< g^*$, then team x has been eliminated.

Proof of Claim 1.2

- Consider any s - t min-cut (A, B) in the graph.
- Claim 1.2.1: If v_{ij} is in A , then both v_i and v_j are in A .
- Claim 1.2.2: If both v_i and v_j are in A , then v_{ij} is in A .



Network Flow

Team Elimination

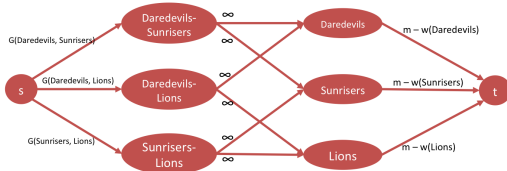
- Claim 1: Team x has been eliminated **iff** the maximum flow in the network is $< g^*$, where $g^* = \sum_{i,j \text{ s.t. } x \notin \{i,j\}} G(i,j)$.

Proof of Claim 1

- Claim 1.1: If x has been eliminated, then the max flow in the network is $< g^*$.
- Claim 1.2: If the max flow is $< g^*$, then team x has been eliminated.

Proof of Claim 1.2

- Consider any s - t min-cut (A, B) in the graph.
- Claim 1.2.1: If v_{ij} is in A , then both v_i and v_j are in A .
- Claim 1.2.2: If both v_i and v_j are in A , then v_{ij} is in A .
- Let T be the set of teams such that $i \in T$ **iff** $v_i \in A$.



Network Flow

Team Elimination

- Claim 1: Team x has been eliminated **iff** the maximum flow in the network is $< g^*$, where $g^* = \sum_{i,j \text{ s.t. } x \notin \{i,j\}} G(i,j)$.

Proof of Claim 1

- Claim 1.1: If x has been eliminated, then the max flow in the network is $< g^*$.
- Claim 1.2: If the max flow is $< g^*$, then team x has been eliminated.

Proof of Claim 1.2

- Consider any s - t min-cut (A, B) in the graph.
- Claim 1.2.1: If v_{ij} is in A , then both v_i and v_j are in A .
- Claim 1.2.2: If both v_i and v_j are in A , then v_{ij} is in A .
- Let T be the set of teams such that $i \in T$ **iff** $v_i \in A$. Then we have:

$$\begin{aligned} C(A, B) &= \sum_{i \in T} (m - w(i)) + \sum_{\{i,j\} \not\subset T} G(i,j) < g^* \\ \Rightarrow m \cdot |T| - \sum_{i \in T} w(i) + (g^* - \sum_{\{i,j\} \subset T} G(i,j)) &< g^* \\ \Rightarrow \sum_{i \in T} w(i) + \sum_{\{i,j\} \subset T} G(i,j) &> m \cdot |T| \quad \square \end{aligned}$$



Applications of Network Flow

Network Flow

Image Segmentation

- You are given an image as a 2-D matrix of pixels.
- We want to determine the foreground and the background pixels.
- Each pixel i , has an integer $a(i)$ associated with it denoting how likely it is to be a foreground pixel.
- Similarly, each pixel i , has an integer $b(i)$ associated with it denoting how likely it is to be a background pixel.
- For neighboring pixels, i and j , there is an associated penalty $p(i,j)$ with putting i and j in different sets.

Problem

Find a partition of the pixels into F and B such that:

$$\sum_{i \in F} a(i) + \sum_{i \in B} b(i) - \sum_{i \text{ and } j \text{ are neighbors but in different sets}} p(i,j)$$

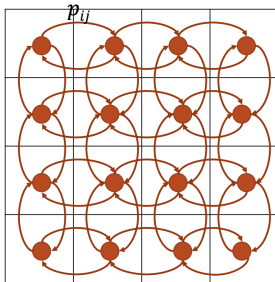
is maximized.

Problem

Find a partition of the pixels into F and B such that:

$$\sum_{i \in F} a(i) + \sum_{i \in B} b(i) - \sum_{i \text{ and } j \text{ are neighbors but in different sets}} p(i, j)$$

is maximized.



Network Flow

Image Segmentation

Problem

Find a partition of the pixels into F and B such that:

$$\sum_{i \in F} a(i) + \sum_{i \in B} b(i) - \sum_{i \text{ and } j \text{ are neighbors but in different sets}} p(i, j)$$

is maximized.

- Consider the network below:

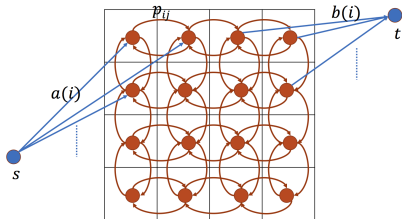


Figure: Idea: The s - t min-cut in the above network gives the optimal partition.

Network Flow

Image Segmentation

- Let $C = \sum_i a(i) + \sum_i b(i)$.
- Claim 1: Consider a partition (F, B) of the set of pixels. Let $S = F \cup \{s\}$, $T = B \cup \{t\}$. Then the capacity of the s - t cut (S, T) in the network is given by

$$C(S, T) = C - \left(\sum_{i \in F} a(i) + \sum_{i \in B} b(i) - \sum_{\substack{i \text{ and } j \text{ are neighbors} \\ \text{but in different sets}}} p(i, j) \right)$$

Network Flow

Image Segmentation

- Let $C = \sum_i a(i) + \sum_i b(i)$.
- Claim 1: Consider a partition (F, B) of the set of pixels. Let $S = F \cup \{s\}$, $T = B \cup \{t\}$. Then the capacity of the s - t cut (S, T) in the network is given by

$$C(S, T) = C - \left(\sum_{i \in F} a(i) + \sum_{i \in B} b(i) - \sum_{\substack{i \text{ and } j \text{ are neighbors} \\ \text{but in different sets}}} p(i, j) \right)$$

- Claim 2: Consider an s - t cut (S, T) in the network. Let $F = A \setminus \{s\}$, $B = T \setminus \{t\}$. Then

$$C(S, T) = C - \left(\sum_{i \in F} a(i) + \sum_{i \in B} b(i) - \sum_{\substack{i \text{ and } j \text{ are neighbors} \\ \text{but in different sets}}} p(i, j) \right)$$

Network Flow

Image Segmentation

- Let $C = \sum_i a(i) + \sum_i b(i)$.
- Claim 1: Consider a partition (F, B) of the set of pixels. Let $S = F \cup \{s\}$, $T = B \cup \{t\}$. Then the capacity of the s - t cut (S, T) in the network is given by

$$C(S, T) = C - \left(\sum_{i \in F} a(i) + \sum_{i \in B} b(i) - \sum_{\substack{i \text{ and } j \text{ are neighbors} \\ \text{but in different sets}}} p(i, j) \right)$$

- Claim 2: Consider an s - t cut (S, T) in the network. Let $F = A \setminus \{s\}$, $B = T \setminus \{t\}$. Then

$$C(S, T) = C - \left(\sum_{i \in F} a(i) + \sum_{i \in B} b(i) - \sum_{\substack{i \text{ and } j \text{ are neighbors} \\ \text{but in different sets}}} p(i, j) \right)$$

- From Claims 1 and 2, we get that if (S, T) is a s - t min-cut in the network, then $F = S \setminus \{s\}$, $B = T \setminus \{t\}$ is an optimal solution to the Image Segmentation problem

Network Flow

Project Selection

Problem

There are n projects each associated with a profit $p(i)$ (*this could be positive or negative integer*). There are dependencies between projects. These dependencies are stored using a dependency graph G where there is a directed edge from project i to project j iff project i depends on project j . Find a **feasible subset** A of projects such that $\sum_{i \in A} p(i)$ is maximized.

- Feasible subset: A subset A is called feasible iff for any edge (i, j) , $i \in A \Rightarrow j \in A$.

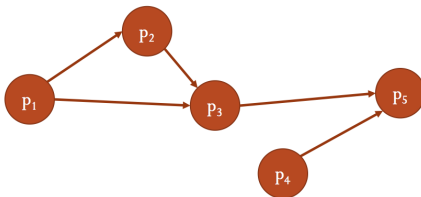


Figure: An example with dependencies.

Network Flow

Project Selection

Problem

There are n projects each associated with a profit $p(i)$ (*this could be positive or negative integer*). There are dependencies between projects. These dependencies are stored using a dependency graph G where there is a directed edge from project i to project j iff project i depends on project j . Find a **feasible subset** A of projects such that $\sum_{i \in A} p(i)$ is maximized.

- Feasible subset: A subset A is called feasible iff for any edge (i, j) , $i \in A \Rightarrow j \in A$.

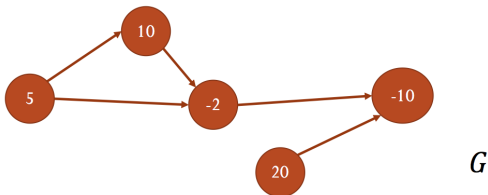


Figure: An example with dependencies and profits.

- Consider the following flow network G' :
 - Add a source s and a sink t .
 - For all i such that $p(i) > 0$, there is an edge (s, i) in G' with capacity $p(i)$.
 - For all i such that $p(i) \leq 0$, there is an edge (i, t) in G' with capacity $-p(i)$.
 - For all edges $(i, j) \in G$, there is an edge (i, j) in G' with capacity ∞ .

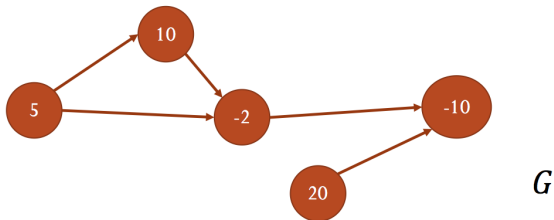
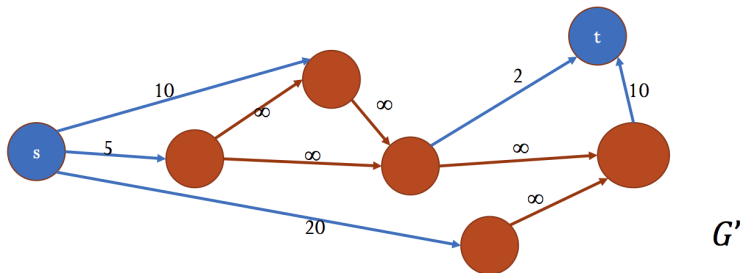


Figure: An example with dependencies and profits.

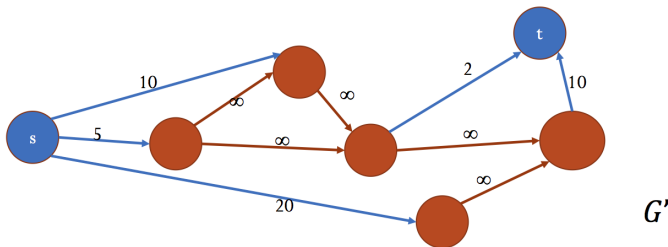
- Consider the following flow network G' :
 - Add a source s and a sink t .
 - For all i such that $p(i) > 0$, there is an edge (s, i) in G' with capacity $p(i)$.
 - For all i such that $p(i) \leq 0$, there is an edge (i, t) in G' with capacity $-p(i)$.
 - For all edges $(i, j) \in G$, there is an edge (i, j) in G' with capacity ∞ .



Network Flow

Project Selection

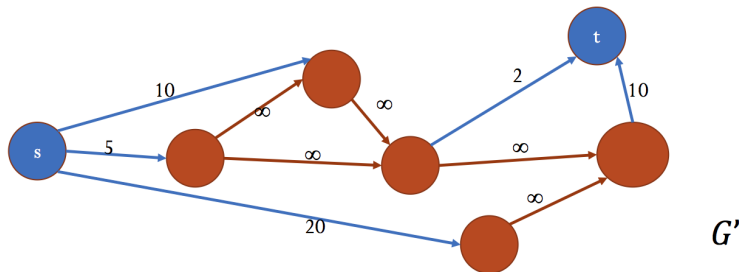
- Let $C = \sum_{i \text{ s.t. } p(i) > 0} p(i)$.
- Claim 1: For any feasible subset A , there is an s - t cut (A', B') in G' such that $c(A', B') = C - \sum_{i \in A} p(i)$.



Network Flow

Project Selection

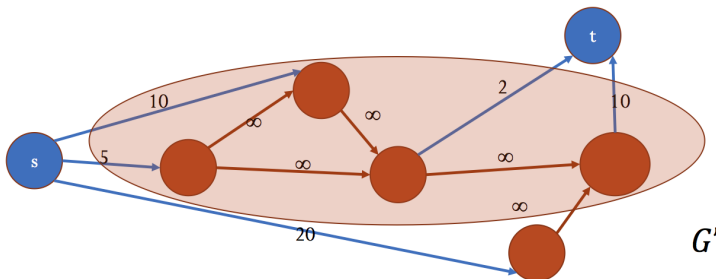
- Let $C = \sum_{i \text{ s.t. } p(i) > 0} p(i)$.
- Claim 1: For any feasible subset A , there is an s - t cut (A', B') in G' such that $c(A', B') = C - \sum_{i \in A} p(i)$.
 - Proof sketch: Consider $A' = A \cup \{s\}$.



Network Flow

Project Selection

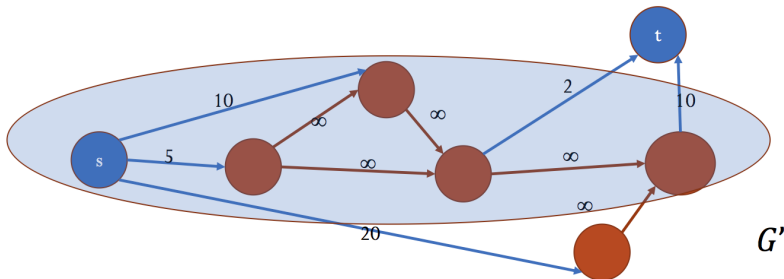
- Let $C = \sum_i \text{s.t. } p(i) > 0 p(i)$.
- Claim 1: For any feasible subset A , there is an s - t cut (A', B') in G' such that $c(A', B') = C - \sum_{i \in A} p(i)$.
 - Proof sketch: Consider $A' = A \cup \{s\}$.



Network Flow

Project Selection

- Let $C = \sum_i \text{s.t. } p(i) > 0 p(i)$.
- Claim 1: For any feasible subset A , there is an s - t cut (A', B') in G' such that $c(A', B') = C - \sum_{i \in A} p(i)$.
 - Proof sketch: Consider $A' = A \cup \{s\}$.

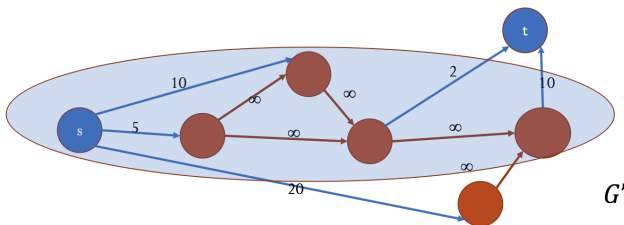


Network Flow

Project Selection

- Let $C = \sum_i \text{s.t. } p(i) > 0 \ p(i)$.
- Claim 1: For any feasible subset A , there is an s - t cut (A', B') in G' such that $c(A', B') = C - \sum_{i \in A} p(i)$.
 - Proof sketch: Consider $A' = A \cup \{s\}$. We have:

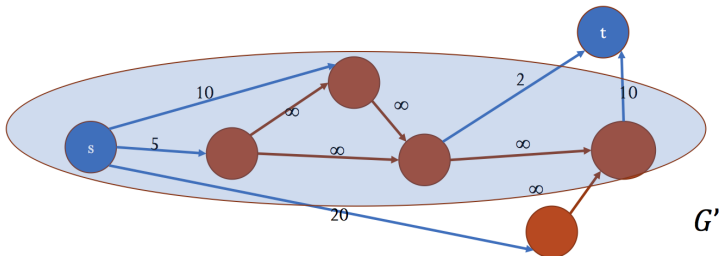
$$\begin{aligned} c(A', B') &= \sum_{i \notin A, p(i) > 0} p(i) - \sum_{i \in A, p(i) \leq 0} p(i) \\ &= C - \sum_{i \in A, p(i) > 0} p(i) - \sum_{i \in A, p(i) \leq 0} p(i) \\ &= C - \sum_{i \in A} p(i) \end{aligned}$$



Network Flow

Project Selection

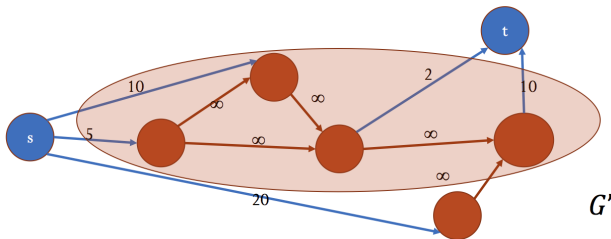
- Let $C = \sum_{i \text{ s.t. } p(i) > 0} p(i)$.
- Claim 1: For any feasible subset A , there is an s - t cut (A', B') in G' such that $c(A', B') = C - \sum_{i \in A} p(i)$.
- Claim 2: For any s - t cut (A', B') in G' of capacity at most C , $A = A' \setminus \{s\}$ is a feasible subset. Moreover, $c(A', B') = C - \sum_{i \in A} p(i)$.



Network Flow

Project Selection

- Let $C = \sum_i \text{s.t. } p(i) > 0 p(i)$.
- Claim 1: For any feasible subset A , there is an s - t cut (A', B') in G' such that $c(A', B') = C - \sum_{i \in A} p(i)$.
- Claim 2: For any s - t cut (A', B') in G' of capacity at most C , $A = A' \setminus \{s\}$ is a feasible subset. Moreover, $c(A', B') = C - \sum_{i \in A} p(i)$.
 - Proof sketch: Since all edges (i, j) corresponding to G have capacity ∞ .



- Let $C = \sum_{i \text{ s.t. } p(i) > 0} p(i)$.
- Claim 1: For any feasible subset A , there is an s - t cut (A', B') in G' such that $c(A', B') = C - \sum_{i \in A} p(i)$.
- Claim 2: For any s - t cut (A', B') in G' of capacity at most C , $A = A' \setminus \{s\}$ is a feasible subset. Moreover, $c(A', B') = C - \sum_{i \in A} p(i)$.
- Claim 3: If (A', B') is the minimum cut in G' , then $A' \setminus \{s\}$ is an optimal solution to the project selection problem.

End